

- 📖

BLOG

▼
- 🤖

LIBRERÍA ANDROID
- ★

COMIENZA AQUÍ
- 📖

BLOG

▼
- 🤖

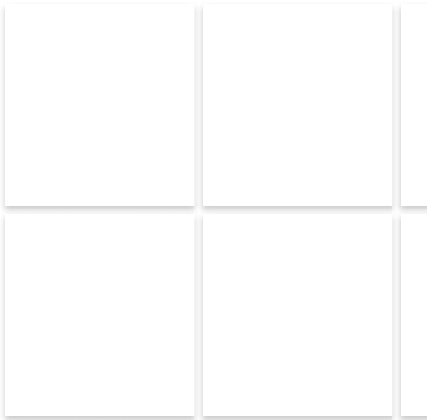
LIBRERÍA ANDROID
- ★

COMIENZA AQUÍ

¿Cómo Obtener La Ubicación De Tus Usuarios En Android?

 [James Revelo](#)  Agosto 26, 2016

“Quiero crear una app que **monitoree la ubicación** del usuario y **detecte su actividad**”



¿Ese tu caso?

Si tu respuesta es afirmativa, entonces llegaste al tutorial correcto.

Con todo el auge de los dispositivos móviles en el mundo y la continua evolución de Android, los usuarios esperan obtener las mejores experiencias asociadas a su accionar diario. Entre las más importantes se encuentra esta: La **ubicación**.

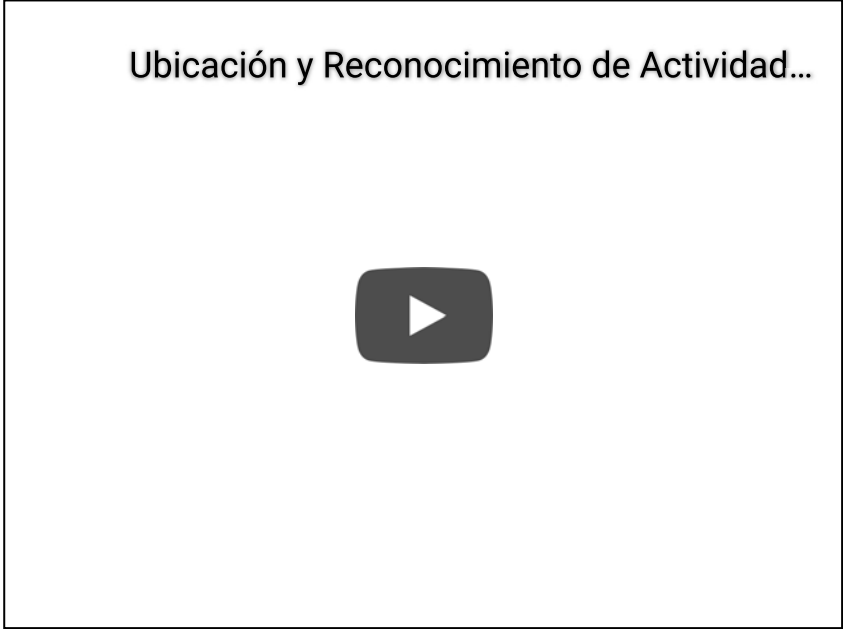
Temas como el **seguimiento de coordenadas, reconocimiento de actividad, Geofencing** hacen parte de muchas apps que llevan a otro nivel la información que proporcionan.

Por esa razón Google nos provee las “Google Play Services location APIs” para manejar la ubicación en Android.



Y he aquí como usarlas.

Primero: Descargar Código De Location Tracker



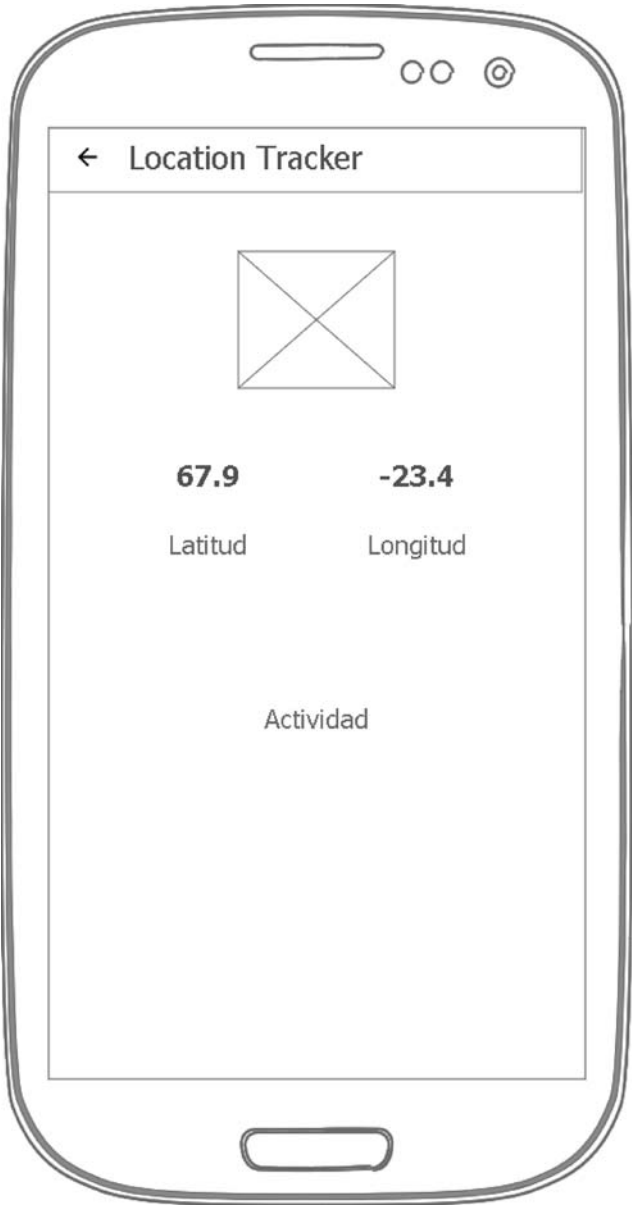
Para tener el resultado final del tutorial de inmediato, puedes suscribirte al boletín informativo de Hermosa Programación para desbloquear el proyecto en Android Studio completo:

<https://www.dropbox.com/s/c6kx24um8276s77/LocationTracker.rar?dl=0>

App Location Tracker

Antes que nos adentremos en la codificación, te explicaré cual será el ejemplo guía de ubicaciones.

Se trata de la aplicación “Location Tracker”.



Obtén Contenido Exclusivo

Aprende a programar apps Android con tips y recursos que solo comparto con los suscriptores de mi boletín privado

Correo

SUSCRIBIRSE

¿Qué buscas?

Buscar

Lo Más Popular

-
- Crear Un Web Service Para Android Con Mysql, Php y Json
-
- ¿Cómo Sincronizar Sqlite Con Mysql En Android?
-
- Aplicación Android Con Navigation Drawer Y Tabs
-
- NavigationView: Navigation Drawer Con Material Design
-
- TabLayout: ¿Cómo Añadir Pestañas En Android?
-
- Realizar Peticiones Http Con La Librería Volley En Android
-
- Tutorial De Bases De Datos SQLite En Aplicaciones Android



Su objetivo será monitorear la ubicación del usuario que instale la app en su teléfono y mostrar en pantalla las coordenadas latitud y longitud.

Adicionalmente mostrará un icono alusivo a la actividad que está realizando.

Ejemplos de ellas son **inactividad**, **caminata**, **movimiento en bicicleta**, **auto**, etc.

¿Suena genial?

Lo es.

Veamos como iniciar...

Obtener La Ubicación Actual Del Usuario

Lo primero que estudiaremos será como obtener la **ubicación actual** del usuario. Aunque también te puedes referir a este concepto como la **última ubicación conocida**.

La pregunta es:

¿Qué clases de las location APIs debes usar?

com.google.android.gms.location

★★★★★

Interfaces

ActivityRecognitionApi	The main entry point for interacting with activity recognition.
FusedLocationProviderApi	The main entry point for interacting with the fused location provider.
Geofence	Represents a geographical region, also known as a geofence.

Según [Google](#), debes **interactuar con el fused location provider** (interfaz **FusedLocationProviderApi**). El componente encargado de obtener las ubicaciones.

Y es que además te facilita las peticiones con poco código. Además te permite controlar la **exactitud y consumo de energía**.

Todo bien hasta aquí, ¿correcto?...

...sin embargo necesitas instalar los servicios de Google antes de adentrarte a la implementación.


Así que te voy a mostrar cómo hacerlo...paso a paso.

Crear proyecto en Android Studio

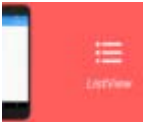
Para comenzar abre [Android Studio](#), crea un nuevo proyecto llamado "Location Tracker" donde,

- El **SDK mínimo** sea 13






Servicio Web RESTful Para Android Con Php, Mysql y Json



Tutorial De Listas Y Adaptadores En Android



Toolbar En Android: Creación De Action Bar En Material Design

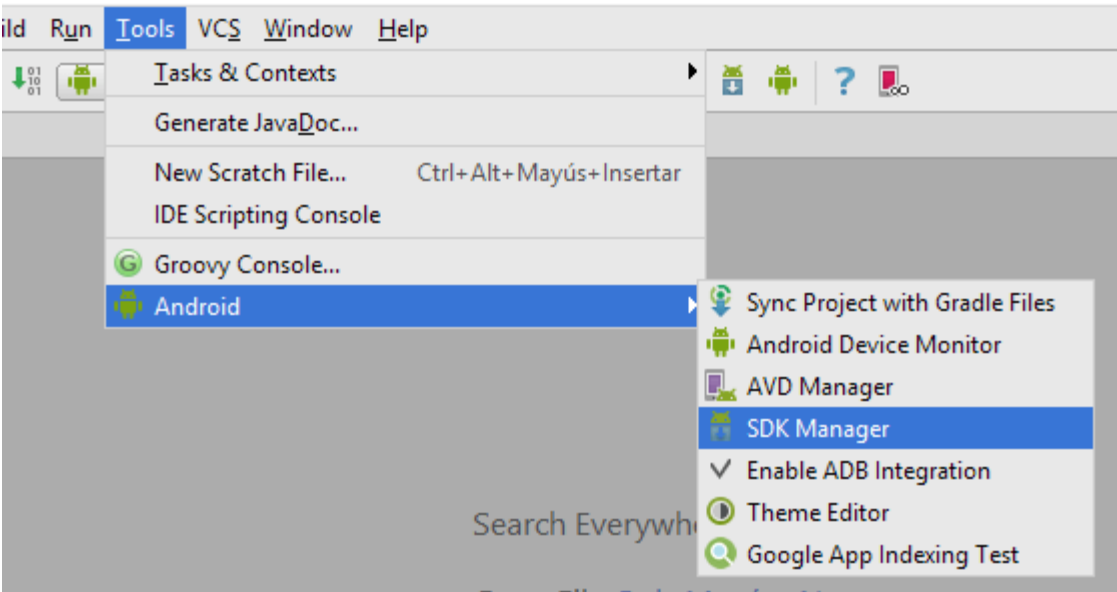
- La actividad inicial sea tipo **Basic Activity**
- La actividad se llame “LocationActivity”

Instalar y Configurar Google Play Services

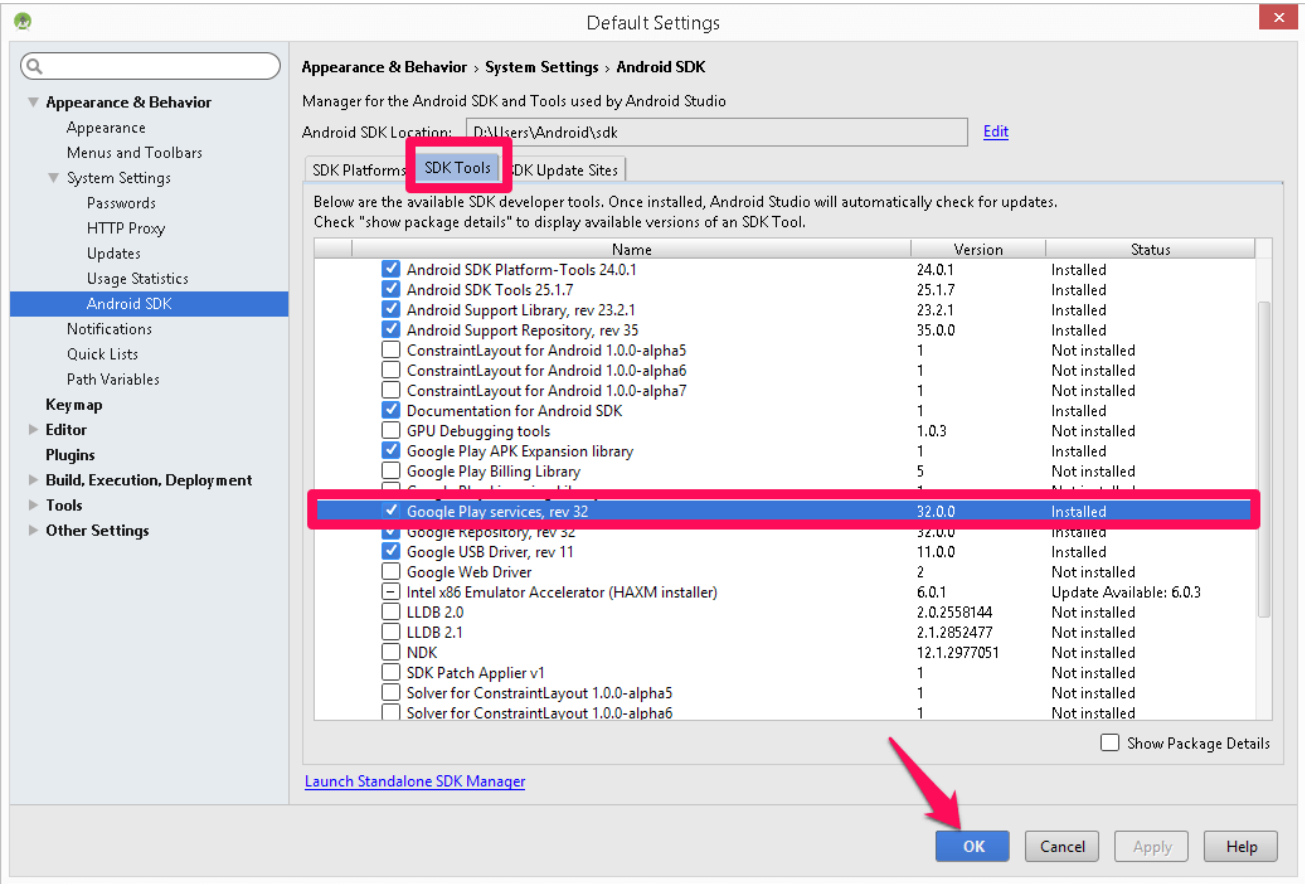
Ahora es tiempo de abrir el SDK Manager e instalar Google Play Services.

¿Cómo?

Ve a **Tools > Android > SDK Manager**.



De inmediato pasa a la pestaña **SDK Tools**. Busca el ítem **Google Play Services**. Marcalo y presiona **OK**.



Y ya está. Sigue el proceso de instalación confirmado la licencia.

Añadir dependencia de locations en build.gradle

Ahora vamos a añadir Google Play Services locations al proyecto.

Abre tu archivo **build.gradle** a nivel de módulo. Sólo tienes el módulo app. Así que no será difícil encontrarlo.

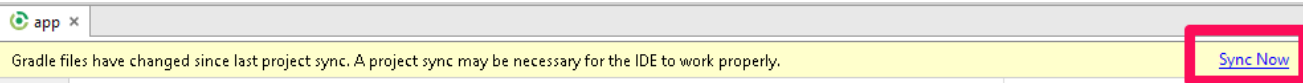


Esta será dependencia que debes agregar.

```
dependencies {  
    compile 'com.google.android.gms:play-services-location:9.4.0'  
}
```

Sincronizar el proyecto

Cuando alteras los archivos de Gradle este mensaje es mostrado.



Esto significa que es necesario sincronizar el proyecto. Para ello presiona **Sync Now**.

Una vez hayas integrado Google Play Services Location con tu proyecto, estás listo para obtener ubicaciones.

Agregar permisos para ubicación

Necesitas especificar permisos para que tu app acceda a los componentes de ubicación.

¿Cuáles es la constante que debes añadir en el manifest?

`ACCESS_COARSE_LOCATION` o `ACCESS_FINE_LOCATION`.

¿Cuál es la diferencia?

La precisión. **“fine” es mucho más exacta que “coarse”**.

¿Qué tanto?

Coarse retorna valores aproximados a una manzana. *Fine* da en el punto exacto, pero a costas de un alto consumo de energía.

Ahora abre el archivo **AndroidManifest.xml** y agrega la siguiente línea.

```
<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
    package="com.hermosaprogramacion.locationtracker"  
  
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>  
  
    ...  
</manifest>
```

Diseñar interfaz gráfica de Location Tracker

Ahora es tiempo de diseñar la vista de única actividad que tenemos.

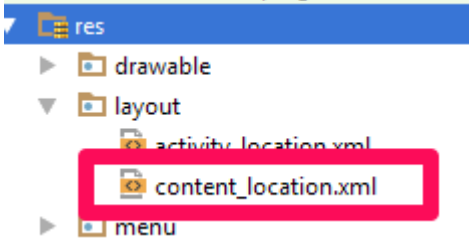
¿En qué archivo debes hacerlo?

Esta parte puede ser confusa si eres principiante.



Y es que Android Studio genera un archivo adicional al crear una actividad básica. Es decir, normalmente tendrás el archivo del layout como `<nombre_actividad>_activity.xml`.

De un tiempo para acá, el contenido principal de este layout se extrae en otro archivo con nombre tipo **`content_<nombre_actividad>.xml`**.



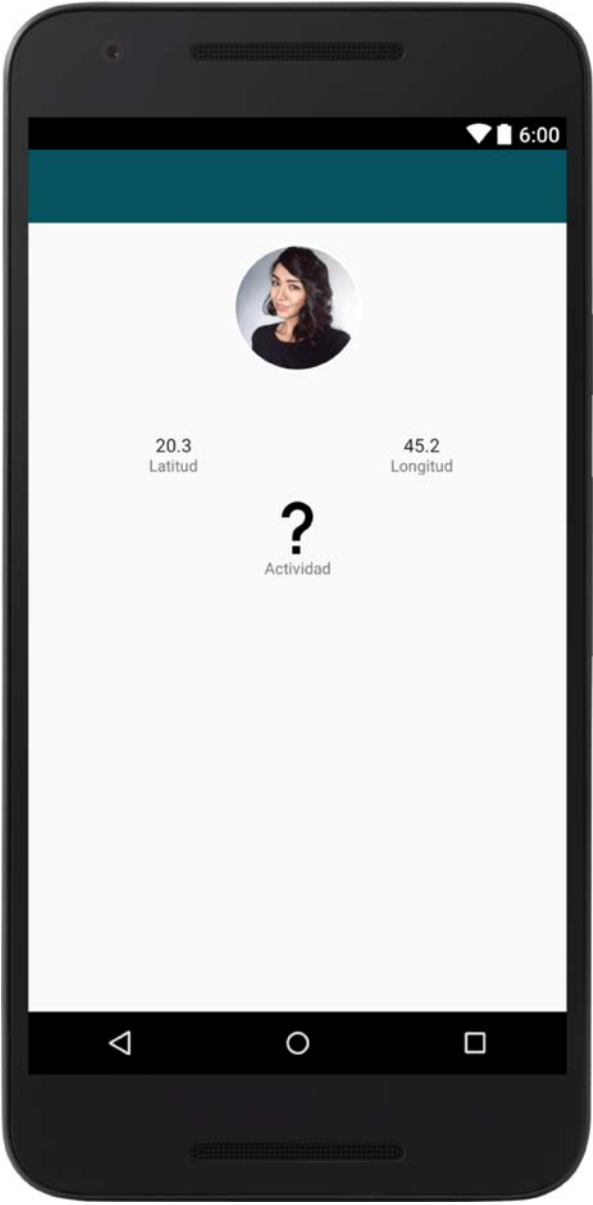
Este será el contenido principal de la actividad.

Y es buena idea ya que separas varios elementos de rutina ([Toolbar](#) y [Navigation Drawer](#)).

Retomando...

Abre el archivo `content_location.xml` y diseña lo que viste en el wireframe inicial. Primero intenta conseguir tu propia versión.

Fíjate en el siguiente mock de alto nivel:



¿Lo tienes?

Bien. Compáralo con el mío para ver qué características puedes mejorar o recomendar.

content_location.xml





```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="com.hermosaprogramacion.locationtracker.LocationActivity"
    tools:showIn="@layout/activity_location">

    <de.hdodenhof.circleimageview.CircleImageView
        android:id="@+id/iv_avatar"
        android:layout_width="98dp"
        android:layout_height="98dp"
        android:layout_centerHorizontal="true"
        android:src="@drawable/avatar_example"
        app:civ_border_color="#FFF"
        app:civ_border_width="2dp" />

    <TableLayout
        android:id="@+id/coordinates"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/iv_avatar"
        android:layout_marginTop="48dp">

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:id="@+id/tv_latitude"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_column="0"
                android:layout_weight="1"
                android:gravity="center_horizontal"
                android:text="New Text"
                android:textAppearance="@style/TextAppearance.AppCompat.Body1"
                tools:text="20.3" />

            <TextView
                android:id="@+id/tv_longitude"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_column="2"
                android:layout_weight="1"
                android:gravity="center_horizontal"
                android:text="New Text"
                android:textAppearance="@style/TextAppearance.AppCompat.Body1"
                tools:text="45.2" />

        </TableRow>

        <TableRow

            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:id="@+id/tv_latitude_label"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_column="0"
                android:layout_weight="1"
                android:gravity="center_horizontal"
                android:text="@string/latitude_label"
                android:textAppearance="@style/TextAppearance.AppCompat.Caption" />

            <TextView
                android:id="@+id/tv_longitude_label"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_column="2"
                android:layout_weight="1"
                android:gravity="center_horizontal"
                android:text="@string/longitude_label"
                android:textAppearance="@style/TextAppearance.AppCompat.Caption" />

        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:gravity="center_horizontal">

            <ImageView
                android:id="@+id/iv_activity_icon"
                android:layout_width="48dp"
                android:layout_height="48dp"
                android:layout_marginTop="@dimen/activity_vertical_margin"
                android:src="@drawable/ic_question" />

        </TableRow>
```



```
<TextView
    android:id="@+id/tv_activity_label"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:gravity="center_horizontal"
    android:text="@string/activity_label"
    android:textAppearance="@style/TextAppearance.AppCompat.Caption" />

</TableLayout>

</RelativeLayout>
```

ImageView circular: La clase `CircleImageView` me permite redondear el avatar. Naturalmente que es necesario incluir la [siguiente dependencia](#):

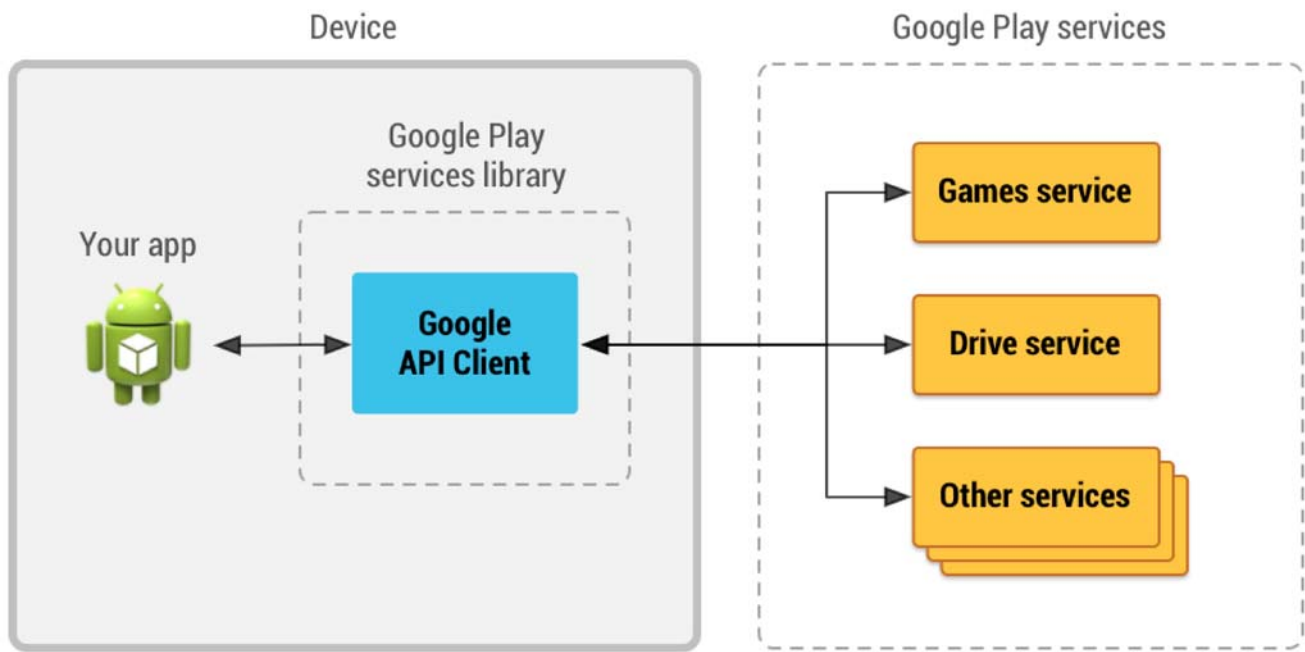
```
compile 'de.hdodenhof:circleimageview:2.1.0'
```

Implementar interfaces de conexión y fallas

[Google dice](#) que quieres acceder a sus servicios, entonces debes usar el componente `GoogleApiClient`.

Y es lógico...

...él es el puente de comunicación entre tu app y los Google Services.



Ahora... ¿cómo crear la conexión a la API de ubicación?

¡Fácil!

Abre `LocationActivity` . Luego implementa sobre ella las interfaces:

- `GoogleApiClient.ConnectionCallbacks` : determina si el cliente está conectado (`onConnected()`) o desconectado (`onConnectionSuspended()`).
- `GoogleApiClient.OnConnectionFailedListener` : procesa los posibles errores de conexión (`onConnectionFailed()`) del cliente al servicio.

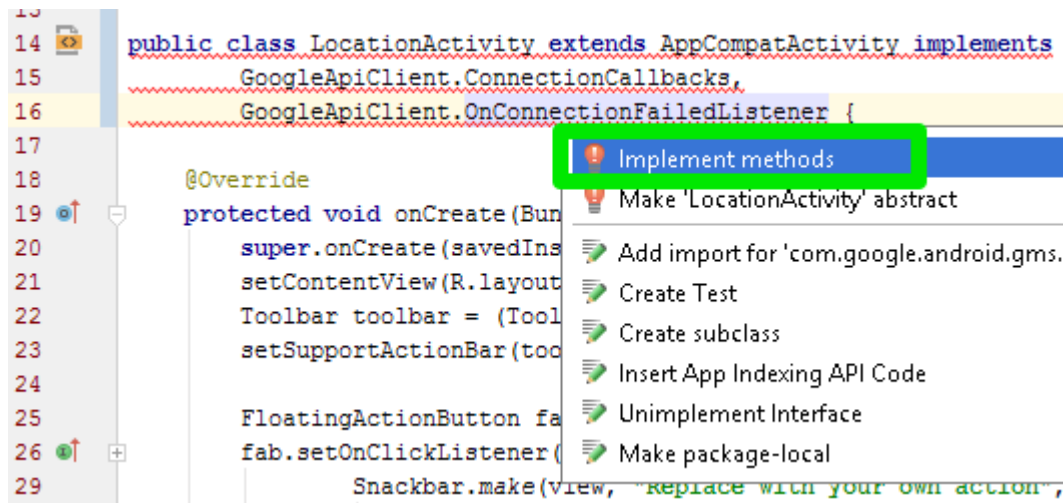
El código parcial sería así:

```
public class LocationActivity extends AppCompatActivity implements
    GoogleApiClient.ConnectionCallbacks,
    GoogleApiClient.OnConnectionFailedListener {
```

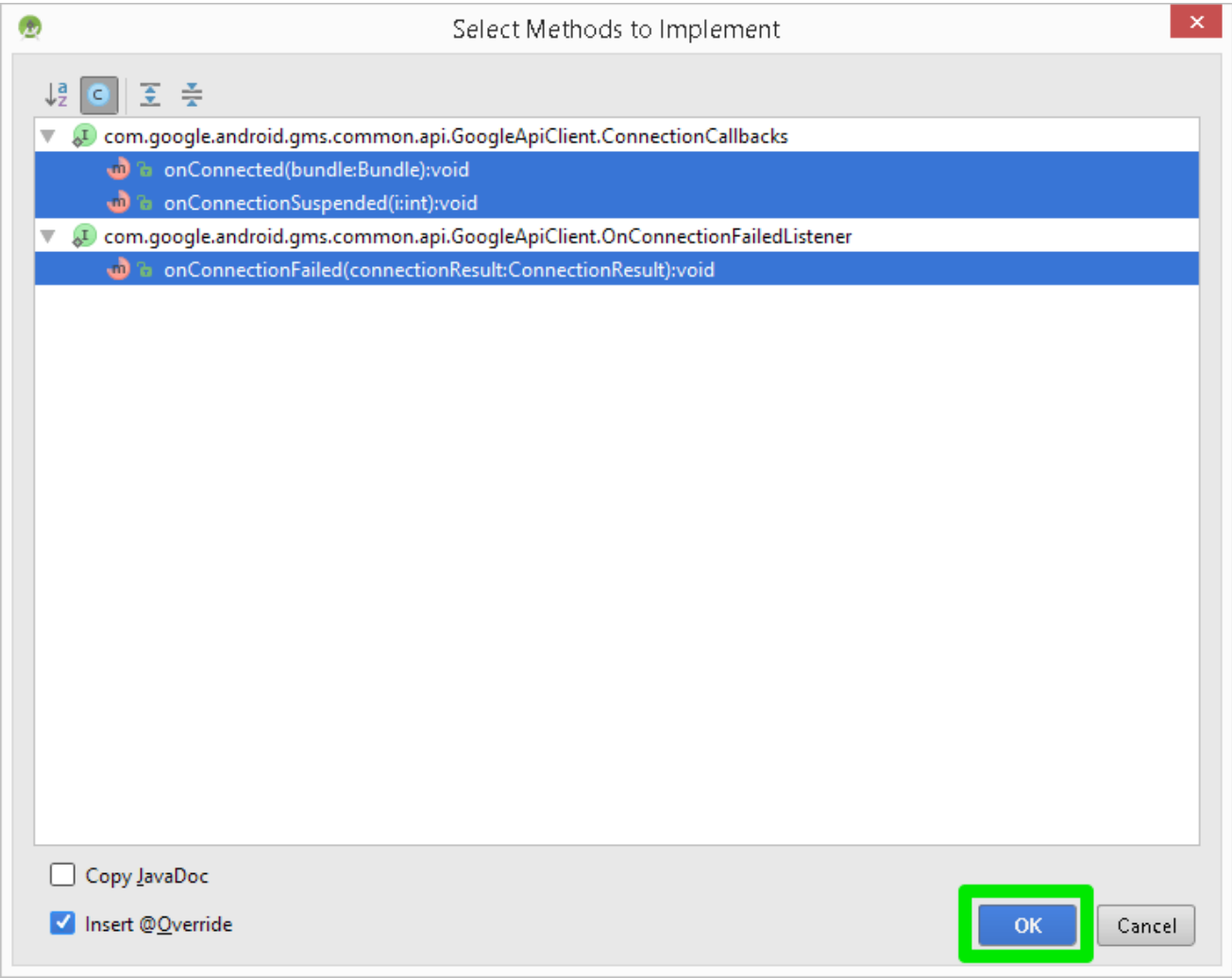

Obviamente verás que Android Studio te alerta de la no implementación de sus controladores.

La solución:

Presiona **alt + insert** y selecciona **Implement methods**.



Ahora vas a ver un cuadro con los métodos que debes sobrescribir. Son obligatorios, así presiona **OK** sin más.



Lo bueno de esta opción es que autogenera el código. Dejando los cuerpos de los métodos escritos...

```
@Override
public void onConnected(@Nullable Bundle bundle) {

}

@Override
public void onConnectionSuspended(int i) {

}

@Override
public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {

}
```

Conectar con Google Play Services

A continuación, ubícate en `onCreate()` y crea la instancia de `GoogleApiClient` .

¿Cuál es la forma de hacerlo?

Aquí está la forma exacta de conectar la API:

Crea un campo tipo `GoogleApiClient` en la actividad. Este alcance te será de utilidad más adelante. La razón es que debes conectar y desconectar la API en `onStart()` y `onStop()` .

```
private GoogleApiClient mGoogleApiClient;
```

Ahora en `onCreate()` usa el patrón `GoogleApiClient.Builder` para generar la instancia.

Los addings que debes usar son:

- La instancia de la interfaz de conexión
- La instancia de la interfaz de fallas
- Y la constante `LocationServices.API`

Así tendrás:

```
// Establecer punto de entrada para La API de ubicación
mGoogleApiClient = new GoogleApiClient.Builder(this)
    .addConnectionCallbacks(this)
    .addOnConnectionFailedListener(this)
    .addApi(LocationServices.API)
    .build();
```

Pero...aún no estás listo.

Lo siguiente es que hagas efectivo el ciclo conexión|desconexión en `onStart()` y `onStop()` .

¿Como?

Bueno, llama los métodos que hagan ambas actividades. `connect()` y `disconnect()` :

```
@Override
protected void onStart() {
    super.onStart();
    mGoogleApiClient.connect();
}

@Override
protected void onStop() {
    super.onStop();
    mGoogleApiClient.disconnect();
}
```

Saber la última ubicación del dispositivo Android

Ahora lo que todos esperábamos.

“Obtener la ubicación actual del usuario”



¡Ah!, pero antes obtén las referencias de los text views que mostrarán la **latitud y altitud**.

De otra mano está la pregunta:

¿Qué método obtiene la ubicación y que clase representa la ubicación?

Es aquí donde Google responde: `LocationServices.FusedLocationApi.getLastLocation()` y `Location`.

Sabiendo esto...posicónate en `onConnected()` y haz la respectiva llamada:

```
@Override
public void onConnected(@Nullable Bundle bundle) {

    mLastLocation = LocationServices.FusedLocationApi.getLastLocation(mGoogleApiClient);
    if (mLastLocation != null) {
        mLatitude.setText(String.valueOf(mLastLocation.getLatitude()));
        mLongitude.setText(String.valueOf(mLastLocation.getLongitude()));
    } else {
        Toast.makeText(this, "Ubicación no encontrada", Toast.LENGTH_LONG).show();
    }
}
```

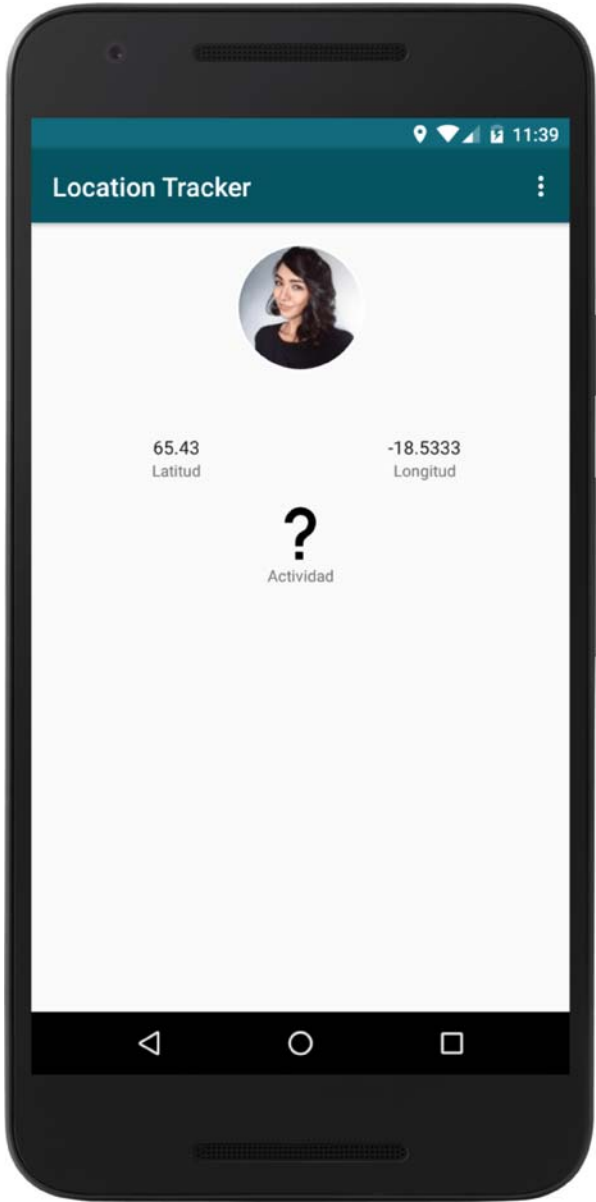
¡Muy bien!

Antes de ejecutar por primera vez la app revisa que...

- 1. Instalaste los servicios de Google Play
- 2. Modificaste las dependencias en **build.gradle**
- 3. Añadiste los permisos de *location coarse*
- 4. Diseñaste la interfaz para mostrar la ubicación
- 5. Implementaste las interfaces de conexión y fallas
- 6. Creaste una instancia de `GoogleApiClient` en `onCreate()`
- 7. Manejaste el ciclo conexión/desconexión en `onStart()` y `onStop()`
- 8. Obtuviste el objeto `Location` en `onConnected()` y mostraste las coordenadas en la vista

Si todo está ready, entonces ejecuta y comprueba.





¿No ves tu ubicación?: es posible que cuando corras la app, no haya una ubicación conocida. Esto se debe que hay un intervalo de lectura de la ubicación mucho mayor al tiempo en que se crea tu actividad. Para ver datos con seguridad, sigue leyendo.

Verificar La Disponibilidad De Google Play Services

Antes que nada comprueba si la APK de Google Play Services está instalada en el dispositivo. También si la versión es compatible con la que usas en tu app.

Hay varias formas que puedes usar para tratar manualmente los errores. Sin embargo en la [guía de acceso a APIs de Google](#) hay algo interesante.

Es posible que usemos el método `enableAutoManage()` para que los errores sean manejados automáticamente.

Y hay más...

...maneja automáticamente la conexión y desconexión de la API de ubicación.

Aquí te dejo el código para la autogestión:

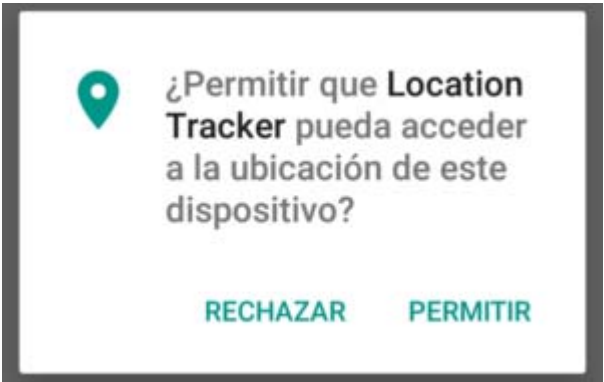
```
// Establecer punto de entrada para La API de ubicación
mGoogleApiClient = new GoogleApiClient.Builder(this)
    .addConnectionCallbacks(this)
    .addOnConnectionFailedListener(this)
    .addApi(LocationServices.API)
    .enableAutoManage(this, this)
    .build();
```

`enableAutoManage()` recibe la actividad que hará la conexión y la interfaz de errores.

Permisos En Tiempo Real Para Google Play Services

¿Programas de **Android Marshmallow** hacia arriba?

No te olvides de asignar *Runtime Permissions*.



Los [permisos en tiempo real](#) se piden si tienes un nivel y target mayor del SDK en versión 23.

Para hacerlo debes...

Verificar la asignación de permisos antes de usar la API de ubicación

Usa la clase `ActivityCompat` para checkear los permisos (`checkSelfPermission()`). Si no están seteados, entonces pídelos (`requestPermissions()`).

Sabiendo esto, ve a `onConnection()` y haz una petición para `ACCESS_FINE_LOCATION` .

```
@Override
public void onConnected(@Nullable Bundle bundle) {
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
        != PackageManager.PERMISSION_GRANTED) {

        if (ActivityCompat.shouldShowRequestPermissionRationale(this,
            Manifest.permission.ACCESS_FINE_LOCATION)) {
            // Aquí muestras confirmación explicativa al usuario
            // por si rechazó los permisos anteriormente
        } else {
            ActivityCompat.requestPermissions(
                this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                REQUEST_LOCATION);
        }
    } else {
        mLastLocation = LocationServices.FusedLocationApi.getLastLocation(mGoogleApiClient);
        if (mLastLocation != null) {
            mLatitude.setText(String.valueOf(mLastLocation.getLatitude()));
            mLongitude.setText(String.valueOf(mLastLocation.getLongitude()));
        } else {
            Toast.makeText(this, "Ubicación no encontrada", Toast.LENGTH_LONG).show();
        }
    }
}
```

Manejar el resultado de la asignación de permisos

La asignación de permisos normalmente es el lanzamiento de un dialogo.

Android le pregunta al usuario algo como:



¿Deseas permitir a esta app que acceda a la ubicación?

El usuario elige, se cierra el diálogo y por lógica debes manejar su respuesta.

La pregunta es:

¿Cómo manejas esta acción?

Usando el método `onRequestPermissionsResult()` .

En él vas a comprobar el código de resultado asignado. Luego ves si los permisos se otorgaron. Y si tuviste respuesta positiva, vuelve a ejecutar la obtención de ubicación.

```
public void onRequestPermissionsResult(int requestCode,
                                     String[] permissions,
                                     int[] grantResults) {
    if (requestCode == REQUEST_LOCATION) {
        if (grantResults.length == 1
            && grantResults[0] == PackageManager.PERMISSION_GRANTED) {

            mLastLocation = LocationServices.FusedLocationApi.getLastLocation(mGoogleApiClient);
            if (mLastLocation != null) {
                mLatitude.setText(String.valueOf(mLastLocation.getLatitude()));
                mLongitude.setText(String.valueOf(mLastLocation.getLongitude()));
            } else {
                Toast.makeText(this, "Ubicación no encontrada", Toast.LENGTH_LONG).show();
            }
        } else {
            Toast.makeText(this, "Permisos no otorgados", Toast.LENGTH_LONG).show();
        }
    }
}
```

Admito que repetir el mismo código de `onConnected()` para mostrar la ubicación actual se ve desorganizado.

Concibamos, pues la creación de métodos para pulverizar las tareas comunes.

Por ejemplo, podemos procesar la obtención de la ubicación y la actualización de la UI así:

```
private void processLastLocation() {
    getLastLocation();
    if (mLastLocation != null) {
        updateLocationUI();
    }
}
```

Donde `getLastLocation()` se conforma de:

```
private void getLastLocation() {
    if (isLocationPermissionGranted()) {
        mLastLocation = LocationServices.FusedLocationApi.getLastLocation(mGoogleApiClient);
    } else {
        manageDeniedPermission();
    }
}

private boolean isLocationPermissionGranted() {
    int permission = ActivityCompat.checkSelfPermission(
        this,
        Manifest.permission.ACCESS_FINE_LOCATION);
    return permission == PackageManager.PERMISSION_GRANTED;
}

private void manageDeniedPermission() {
    if (ActivityCompat.shouldShowRequestPermissionRationale(this,
        Manifest.permission.ACCESS_FINE_LOCATION)) {
        // Aquí muestras confirmación explicativa al usuario
    }
}
```



```
        // por si rechazó los permisos anteriormente
    } else {
        ActivityCompat.requestPermissions(
            this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
            REQUEST_LOCATION);
    }
}
```

Y `updateLocationUI()` es:

```
private void updateLocationUI() {
    mLatitude.setText(String.valueOf(mLastLocation.getLatitude()));
    mLongitude.setText(String.valueOf(mLastLocation.getLongitude()));
}
```

Cambiar Opciones De Ubicación

Mira:

Obtener la ubicación es la acción neurálgica cuando usas el Location Fused Provider. No obstante, es importante que te asegures que las opciones del sistema estén correctamente configuradas.

Por ejemplo, ¿qué tal si el **escaneo del GPS o el Wi-Fi** está desactivado por el usuario?

La petición por ubicaciones jamás tendría un resultado para tu app.

Google nos recomienda usar un diálogo para pedirle al usuario el permiso para habilitar estás opciones.

Veamos cómo hacerlo.

Usando LocationRequest para configurar ubicación

En primer lugar, ya debes tener una conexión a la API.

Que fue justo lo que hicimos arriba.

Ahora, si quieres cambiar la configuración usa la clase `LocationRequest`.

Su función es determinar la calidad del servicio que usarán las peticiones del Fused Location Provider.

¿La solución?

Basándonos en tres características fundamentales de la ubicación en un dispositivo Android:

- 1. Intervalo de actualización
- 2. Intervalo de actualización rápida
- 3. Prioridad

Aquí están los métodos para cada uno:



- `setInterval()` : Establece la frecuencia (en milisegundos) con la que quieres que lleguen actualizaciones a tu app.
- `setFastestInterval()` : Hay aplicaciones que pedirán actualizaciones de ubicación más rápido que la tuya. Estas afectan a tu app y la manera de controlar esta variación es definir un intervalo de lectura menor a `setInterval()` . Así que usa `setFastestInterval()` con el fin de indicar dicho tiempo (milisegundos).
- `setPriority()` : Determina los recursos de ubicación a usarse (GPS, Wi-Fi, Red móvil).

Mira los valores que soporta este método:

Esto solo ocurre cuando otras apps hacen uso de la API de ubicación. Claro está que tendrás un increíble ahorro de energía.

<code>PRIORITY_BALANCED_POWER_ACCURACY</code>	Con él solicitas precisión de una manzana (aprox. una manzana). Además el consumo de batería es menor al usar el Wi-Fi y la red móvil como fuentes de ubicación.
<code>PRIORITY_HIGH_ACCURACY</code>	Obtiene la ubicación con la mayor precisión posible. Y con todo sentido, ya que se usa el GPS. La desventaja es el alto consumo de batería.
<code>PRIORITY_LOW_POWER</code>	Solicitas precisión a nivel de ciudad (10 kilómetros aprox.). Al ser la menos precisa genera buen ahorro de batería.
<code>PRIORITY_NO_POWER</code>	Tu app recibirá actualizaciones de ubicación solo cuando estén disponibles.

Te voy a mostrar la forma de aplicarlo en Location Tracker:

Abre `LocationActivity` y agrega dos constantes. Una para el intervalo normal y otra para el rápido.

```
public static final long UPDATE_INTERVAL = 1000;
public static final long UPDATE_FASTEST_INTERVAL = UPDATE_INTERVAL / 2;
```

Usa el valor que deseas. Yo por mi parte quiero un segundo de actualización para un monitoreo más notable.

Ahora ve a `onCreate()` y crea una nueva instancia de `LocationRequest` . Pon los tiempos que acabamos de establecer y sumale la prioridad `PRIORITY_HIGH_ACCURACY` .

```
// Crear configuración de peticiones
LocationRequest locationRequest = new LocationRequest()
    .setInterval(UPDATE_INTERVAL)
    .setFastestInterval(UPDATE_FASTEST_INTERVAL)
    .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
```

Aún esta configuración no tendrá efecto...

...si quieres aplicar los parámetros entonces obtén las opciones actuales.

Obtener opciones de ubicación actuales

Pon atención:

La clase `LocationSettingsRequest` especifica los servicios de ubicación que la app usará.

¿Qué significa esto?

Quiere decir que perfila todas tus peticiones de ubicación con la configuración que hayas creado.

O sea que recibirá nuestro objeto `LocationRequest` previamente creado. Y es que realizarlo es sencillo.

Tan solo usa su patrón `Builder` y agrégalo con el método `addLocationRequest()`. Fijate como:

```
// Crear opciones de peticiones
LocationSettingsRequest.Builder builder = new LocationSettingsRequest.Builder()
    .addLocationRequest(locationRequest)
    .setAlwaysShow(true);
```

Ahora comprueba si el usuario tiene configurado los ajustes de ubicación para soportar lo que le pides.

Suena fácil, pero... ¿qué hacer?

Aquí está la respuesta:

Emplea el método `LocationServices.SettingsApi.checkLocationSettings()`. Sus parámetros ya los tienes. El primero es el cliente de la API de Google. Y el segundo es el objeto `LocationSettingsRequest`.

```
// Verificar ajustes de ubicación actuales
PendingResult<LocationSettingsResult> result = LocationServices.SettingsApi.checkLocationSettings(
    mGoogleApiClient, builder.build()
);
```

Lo siguiente es que notes que la respuesta fue un objeto `PendingResult<LocationSettingsResult>`.

En él está todo el meollo del asunto con el usuario.

Así que mantente conmigo y observa porque...

Pedir al usuario cambiar los ajustes de ubicación

El resultado obtenido por el chequeo nos permite saber si debemos pedir al usuario que cambie los ajustes.

En ese orden de cosas asignaremos una callback al objeto con el fin de saber si podemos aplicar las opciones construidas.



La clase `ResultCallback<LocationSettingsResult>` será la que cumpla este cometido. Usa su controlador `onResult()` para comprobar los posibles resultados:

- `SUCCESS` : Los ajustes actuales del usuario permiten que avances satisfactoriamente.
- `RESOLUTION_REQUIRED` : Los ajustes no satisfacen la configuración. Aquí es donde pides al usuario con un diálogo que cambie sus opciones (`startResolutionForResult()`).
- `SETTINGS_CHANGE_UNAVAILABLE` : Los ajustes no satisfacen la configuración. Sin embargo, no es posible hacer nada por resolverlo. Así que no hacemos nada.

Bien, ese es el objeto para el manejo.

Lo siguiente es

Procesar los resultados de ajustes

Usa el método `setResultCallback()` para asignarlo.

Resumiendo nuestro ejemplo, luego de conseguir el resultado de los ajustes, haz esto:

```
result.setResultCallback(new ResultCallback<LocationSettingsResult>() {
    @Override
    public void onResult(@NonNull LocationSettingsResult result) {
        Status status = result.getStatus();

        switch (status.getStatusCode()) {
            case LocationSettingsStatusCodes.SUCCESS:

                break;
            case LocationSettingsStatusCodes.RESOLUTION_REQUIRED:

                break;
            case LocationSettingsStatusCodes.SETTINGS_CHANGE_UNAVAILABLE:

                break;
        }
    }
});
```

A continuación, procesa los **casos de la estructura switch**.

El primero significa que todo ha salido bien. En consecuencia allí realizas la petición de ubicación.

```
case LocationSettingsStatusCodes.SUCCESS:
    Log.d(TAG, "Los ajustes de ubicación satisfacen la configuración.");
    processLastLocation();

    break;
```

En el segundo lanzamos un diálogo de ayuda con `Status.startResolutionForResult()` . Donde pasarás como parámetro la actividad y un código de petición.

```
case LocationSettingsStatusCodes.RESOLUTION_REQUIRED:
    try {
        Log.d(TAG, "Los ajustes de ubicación no satisfacen la configuración. " +
            "Se mostrará un diálogo de ayuda.");
        status.startResolutionForResult(
            LocationActivity.this,
            REQUEST_CHECK_SETTINGS);
    } catch (IntentSender.SendIntentException e) {
        Log.d(TAG, "El Intent del diálogo no funcionó.");
        // Sin operaciones
    }
    break;
```

Y por último, en el tercero, loguea simplemente el resultado adverso.

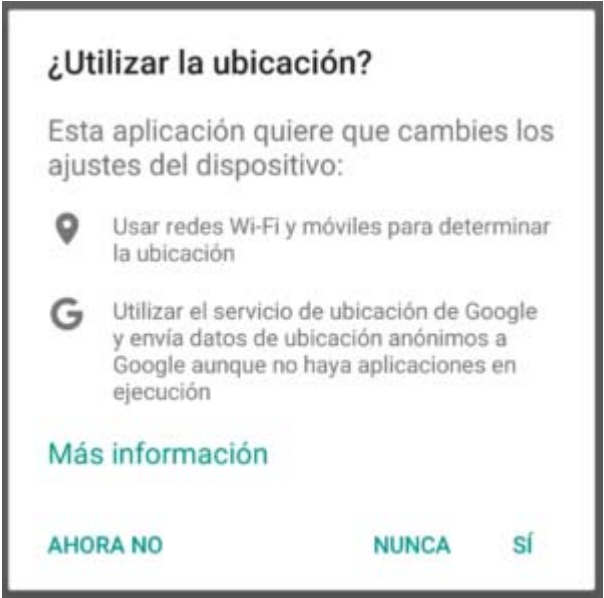
```
case LocationSettingsStatusCodes.SETTINGS_CHANGE_UNAVAILABLE:
    Log.d(TAG, "Los ajustes de ubicación no son apropiados.");
    break;
```

Otra cosa más:

El cierre del diálogo se procesa con `onActivityResult()` . No te olvides de manejar su estado. Si el usuario autorizó el cambio de ajustes, inicia la petición de ubicación.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    switch (requestCode) {
        case REQUEST_CHECK_SETTINGS:
            switch (resultCode) {
                case Activity.RESULT_OK:
                    Log.d(TAG, "El usuario permitió el cambio de ajustes de ubicación.");
                    processLastLocation();
                    break;
                case Activity.RESULT_CANCELED:
                    Log.d(TAG, "El usuario no permitió el cambio de ajustes de ubicación");
                    break;
            }
            break;
    }
}
```

Esto es genial, ya que cada vez que el usuario cambie los ajustes, nuestra app mostrará un mensaje convincente:



Recibir Actualizaciones De Ubicación

Hasta el momento, hemos recibido solo un valor de ubicación. Y eso que si otra app como [Google Maps](#) ha obtenido una actualización antes de abrir Location Traker.

Y eso no es lo que buscamos, ¿correcto?

Necesitamos monitorear constantemente la ubicación. Recibir actualizaciones en intervalos regulares de tiempo.

Afortunadamente para nosotros, Google Play Services Location API trae consigo el método `requestLocationUpdates()` .

A diferencia de `requestLocation()` , `requestLocationUpdates()` consigue actualizaciones según el intervalo definido.



De manera que te mostraré como, cuando y donde llamarlo...

Escuchar nuevas ubicaciones con LocationListener

Cuidado aquí:

Antes de solicitar las actualizaciones debes tener la conexión establecida a la API.

Si te saltaste las secciones pasadas, entonces te recomiendo leerlas. De lo contrario no verás los resultados de este ejercicio.

Por otro lado...

...ubícate en `LocationActivity` y llama dentro de `onConnect()` al método `requestLocationUpdates()`.

El artículo [Receiving Location Updates](#) nos informa que la escucha de actualizaciones es `LocationListener`.

Por esta razón implementala sobre la actividad.

```
public class LocationActivity extends AppCompatActivity implements
    GoogleApiClient.ConnectionCallbacks,
    GoogleApiClient.OnConnectionFailedListener,
    LocationListener{
```

Ahora sobrescribe el controlador `onLocationChanged()`.

Como era de esperarse, este método recibe un objeto `Location`. Por lo que actualizaremos la última ubicación y actualizaremos la interfaz gráfica.

```
@Override
public void onLocationChanged(Location location) {
    Log.d(TAG, String.format("Nueva ubicación: (%s, %s)",
        location.getLatitude(), location.getLongitude()));
    mLastLocation = location;
    updateLocationUI();
}
```

Iniciar obtención de actualizaciones

`requestLocationUpdates()` te pide tres parámetros:

- Una instancia `GoogleApiClient`
- Un objeto `LocationRequest`
- Y la escucha `LocationListener`

Importante:

Esta es solo una variación del método. Úsalo para monitorear en primer plano (*foreground*). Si quieres mantener el tracking en segundo plano (*background*) usa:

```
requestLocationUpdates (GoogleApiClient client, LocationRequest request,
PendingIntent callbackIntent)
```


Teniendo en claro esto, iniciemos el monitoreo en `onConnect()` :

```
@Override
public void onConnected(@Nullable Bundle bundle) {

    // Obtenemos La última ubicación al ser La primera vez
    processLastLocation();
    // Iniciamos Las actualizaciones de ubicación
    startLocationUpdates();
}

private void startLocationUpdates() {
    if (isLocationPermissionGranted()) {
        LocationServices.FusedLocationApi.requestLocationUpdates(
            mGoogleApiClient, mLocationRequest, this);
    } else {
        manageDeniedPermission();
    }
}
```

Por lo general se separa la petición de actualizaciones en un método llamado `startLocationUpdates()` .

¿Cómo Detener Las Actualizaciones De Ubicación?

Ahorra batería del dispositivo frenando las actualizaciones de ubicación si no se necesitan.

La pregunta es:

¿Cuándo no se necesitan?

Al momento de que tu actividad pase a segundo plano.

Por ejemplo, si **se abre otra app** o si **cambias a otra actividad** dentro de la misma aplicación.

El método para detener las updates se llama `removeLocationUpdates()` . Con el fin de optimizar el consumo, llámalo en `onPause()` .

Para que funcione pásale el cliente de Google APIs y la escucha `LocationListener` .

¿Es claro hasta allí?

Bien.

Vamos a ver rápidamente la implementación `removeLocationUpdates()` en nuestro ejemplo:

```
@Override
protected void onPause() {
    super.onPause();
    if (mGoogleApiClient.isConnected()) {
        stopLocationUpdates();
    }
}
```

La [documentación](#) nos sugiere crear el método `stopLocationUpdates()` para simplificar el llamado.



Conservar la transición pausa/reanudación

El ciclo de vida de una actividad frecuentemente es bidireccional.

Así que debemos complementar la detención de actualizaciones con una reanudación.

¿En qué lugar lo hacemos?

En los métodos `onResume()` .

No obstante, comprueba primero si el cliente aún sigue conectado.

```
@Override
protected void onResume() {
    super.onResume();
    if (mGoogleApiClient.isConnected()) {
        startLocationUpdates();
    }
}
```

Observa los cambios del GPS

Para que terminos este punto, ejecuta de nuevo la app.

65.7999983	-19.0
Latitud	Longitud
66.0	-20.0
Latitud	Longitud
67.4	-21.89
Latitud	Longitud
65.48	-18.6999983
Latitud	Longitud

¿Genial, cierto?

El gran problema es el siguiente:

¿Que hacer para generar cambios en el sistema de posicionamiento global?

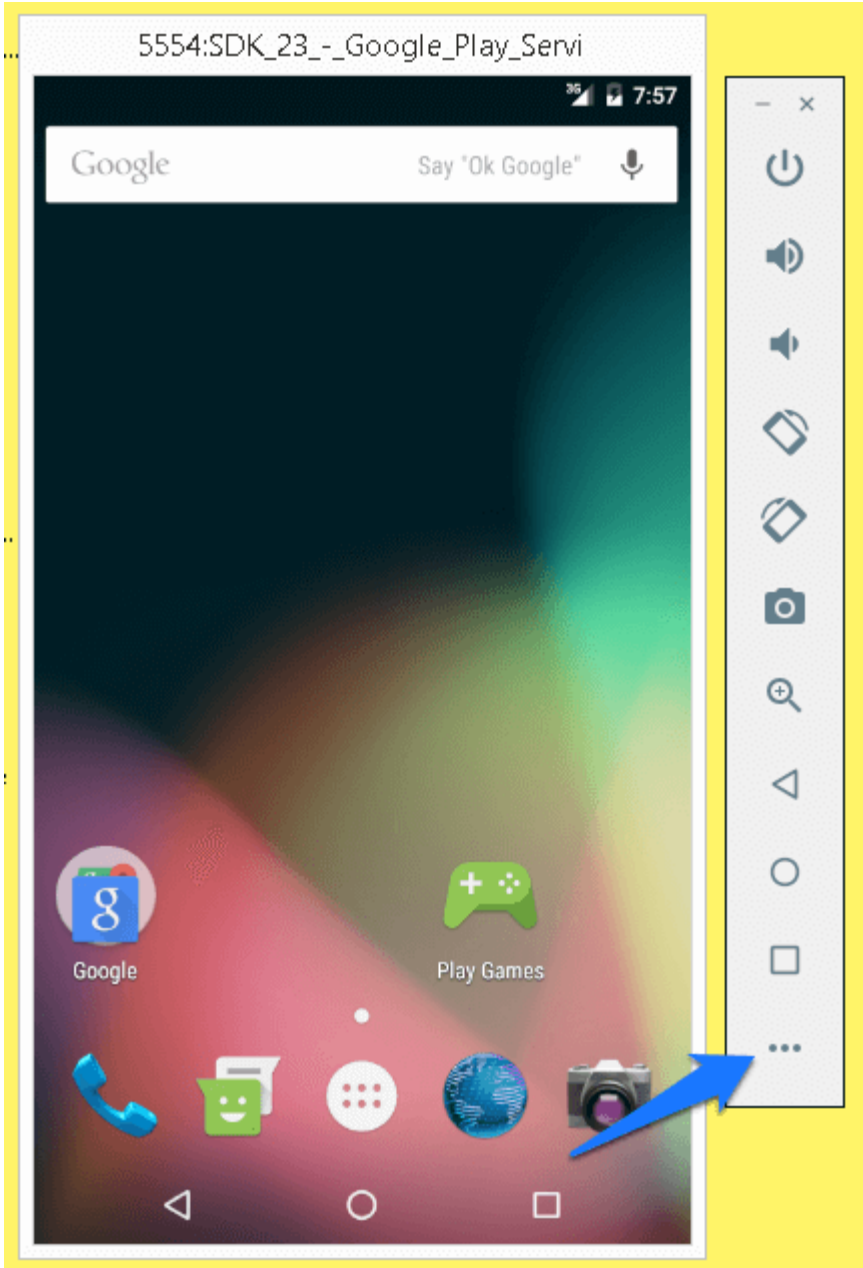
No tiene por qué ser duro para ti.

Aquí te doy la solución.

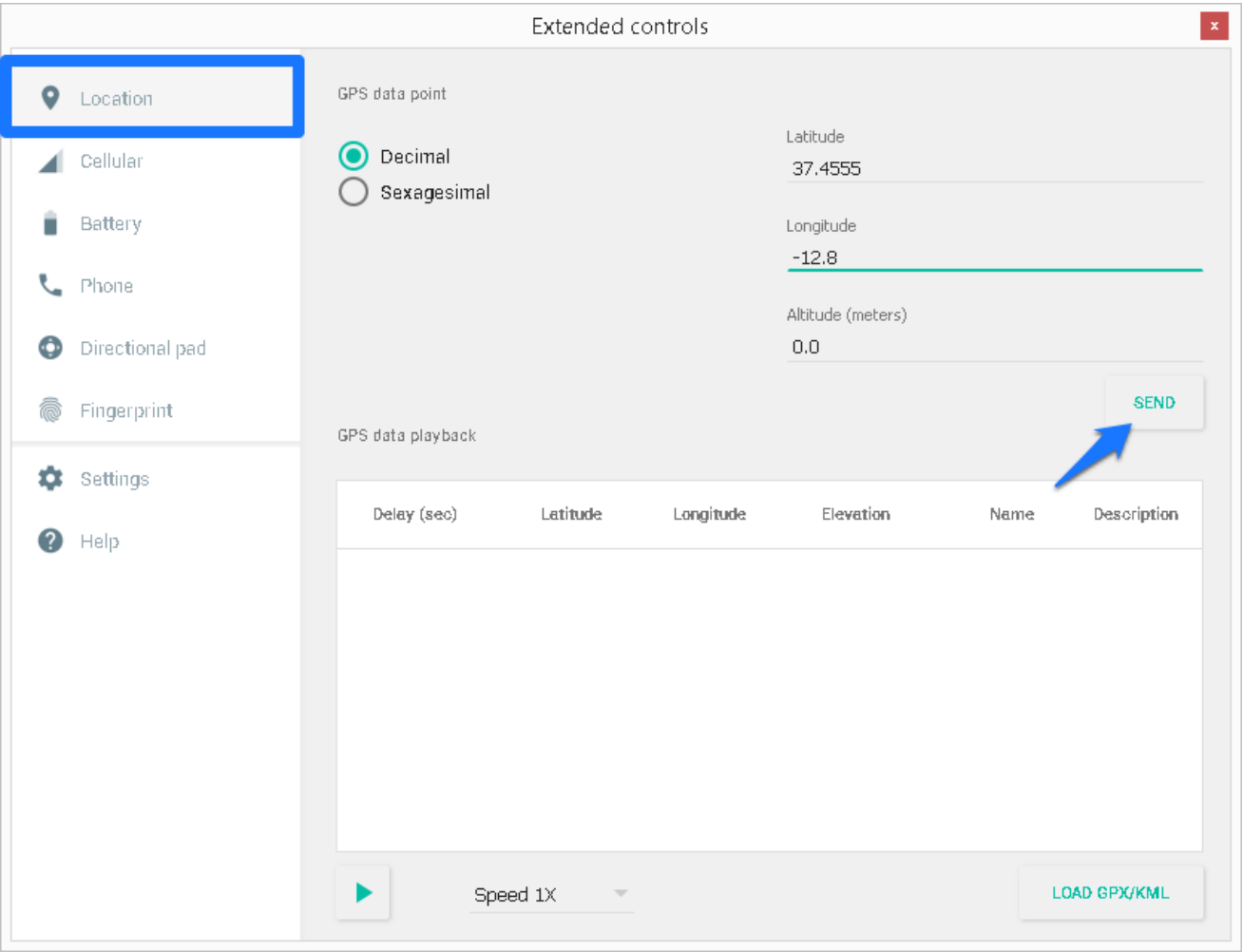
AVD en Android Studio: Ejecuta tu emulador del SDK y corre la app.

Cuándo esté listo, presiona el botón de tres puntos.



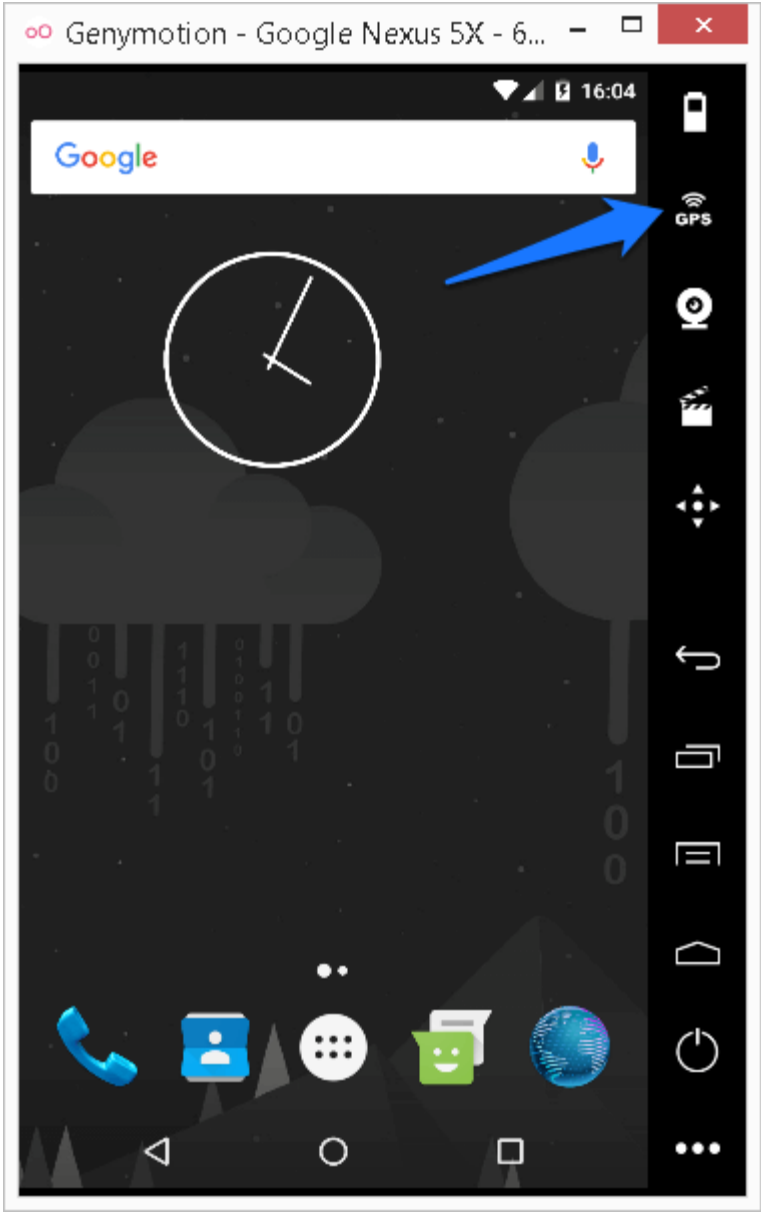


Luego selecciona la sección **Location**.



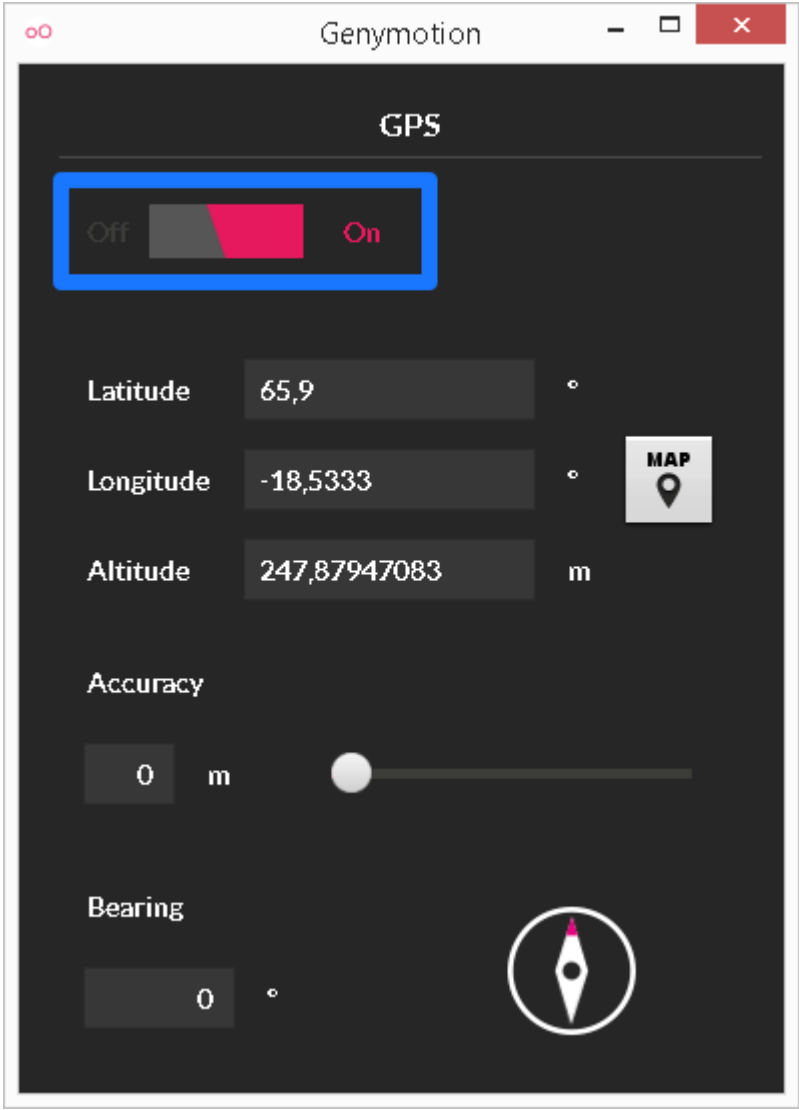
En el panel podrás cambiar los datos actuales del GPS. Cuando hayas preparado sus valores presiona **SEND** y la actualización llegará a la app.

Emuladores de Genymotion: Abre el administrador de Genymotion e inicia tu emulador. Verás un icono con ondas de señales distintivo.



Cambia de **Off** a **On** el estado del GPS.

De esa forma podrás actualizar los datos de ubicación inmediatamente.



No hace falta que confirmes nada. Modifica y cierra el recuadro. El emulador tomará el nuevo valor.

Guardar Ubicación Al Rotar La Pantalla

Si, es importante que guardes el estado de la actividad cuando haya un cambio de configuración.

¿Por qué?

No te gustará que tu actividad se recree y descarte la ubicación que guardamos en `mLastLocation`.

Un giro de pantalla o un cambio de idioma...y ¡zas!, adiós información.

Por eso los [consejos de cambios de configuración](#) nos dicen que debemos sobrescribir `onSaveInstanceState()` para guardar los campos.

Aplicándolo a nuestra app de tracking, tendremos este resultado:

```
@Override
protected void onSaveInstanceState(Bundle outState) {
    // Protegemos La ubicación actual antes del cambio de configuración
    outState.putParcelable(LOCATION_KEY, mLastLocation);
    super.onSaveInstanceState(outState);
}
```

Ahora, retorna el valor en `onCreate()` cuando termine el cambio de configuración. La guía nos recomienda crear un método llamado `updatesValuesFromBundle()` para este objetivo.

Haciéndole caso tendrás:

```
private void updateValuesFromBundle(Bundle savedInstanceState) {
    if (savedInstanceState != null) {
        if (savedInstanceState.containsKey(LOCATION_KEY)) {
            mLastLocation = savedInstanceState.getParcelable(LOCATION_KEY);

            updateLocationUI();
        }
    }
}
```

Donde `containsKey()` comprueba si la clave del valor de ubicación viene en el estado (`Bundle`).

Reconocer Que Actividad Está Realizando El Usuario

Ya para terminar, llegamos a este punto:

Reconocimiento de actividad.

¿Tu usuario está corriendo?

¿Está en un auto?



Hay múltiples estados que puedes detectar a través de la **Activity Recognition API**.

Pienso que es un gran complemento para la detección de ubicaciones. Y más todavía, si deseas tomar decisiones en tu app, basado en lo que realiza el usuario.

Por consiguiente, **armemos un ejemplo**.

Sigue los pasos...

Añadir permisos para reconocimiento de actividad

Esta característica te exige que pongas la siguiente etiqueta de permisos en tu archivo **AndroidManifest.xml**.

```
<uses-permission android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION" />
```

De la otra mano tienes la dependencia gradle. La cual no cambia, ya que el reconocimiento de actividad hace parte del paquete de ubicación. En consecuencia, mantén solo la línea que incluimos al inicio.

```
compile 'com.google.android.gms:play-services-location:9.4.0'
```

Crear IntentService para recibir actualizaciones de actividad

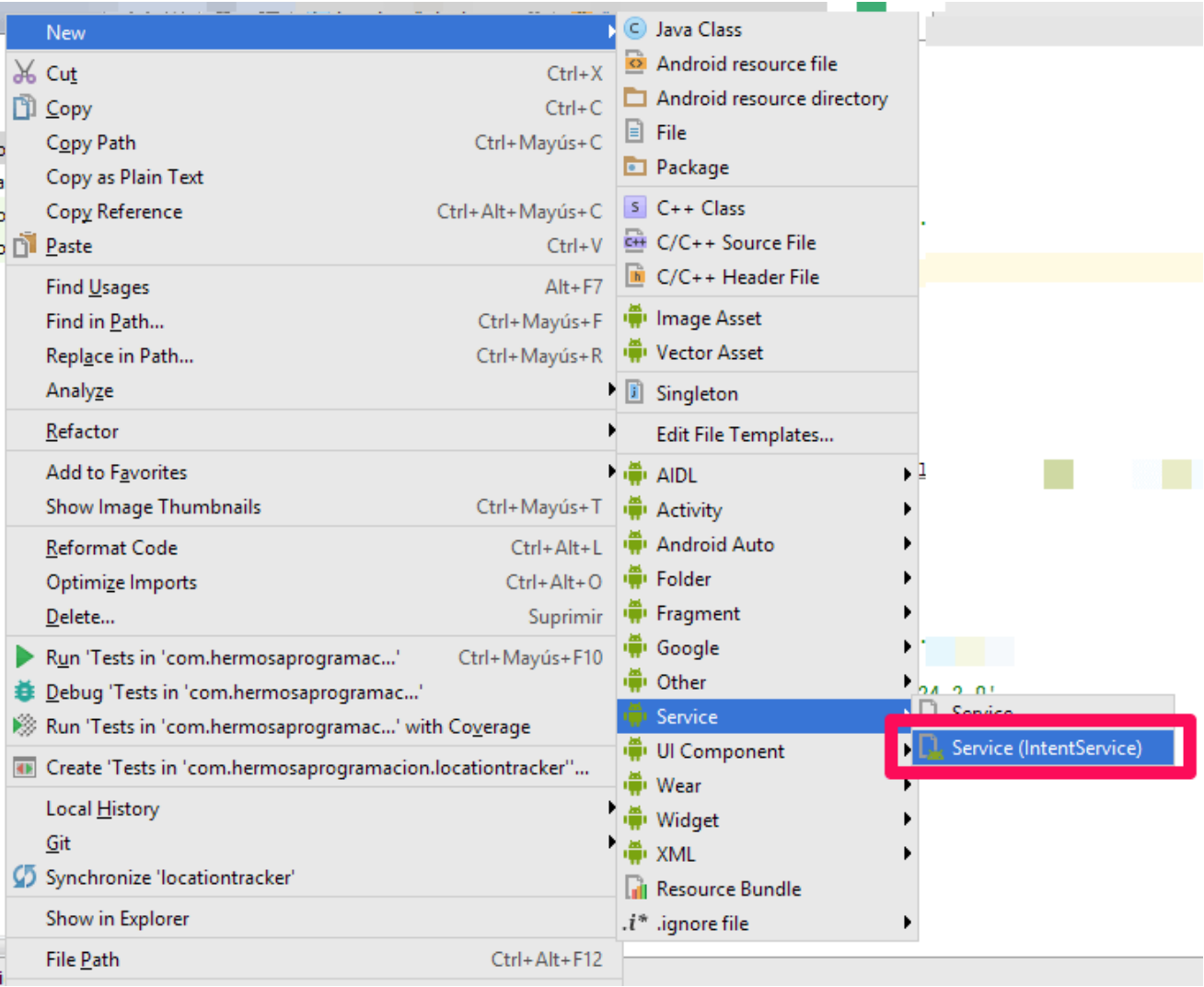
Después de poner el esquema general, haz esto:

[Crea un IntentService](#) para realizar los llamados a la API de actividad en un subproceso de trabajo.

Déjame explicarlo con los menús de Android Studio.

Párate en tu paquete principal Java y presiona click derecho. Ahora selecciona **New > Service > Service (Intent Service)**.





Llámalo `DetectedActivitiesIntentService` y limpia su contenido autogenerado para que tome esta forma:

```
public class DetectedActivitiesIntentService extends IntentService {

    private static final String TAG = DetectedActivitiesIntentService.class.getSimpleName();

    public DetectedActivitiesIntentService() {
        super(TAG);
    }

    @Override
    public void onCreate() {
        super.onCreate();
    }

    @Override
    protected void onHandleIntent(Intent intent) {

    }

}
```

Actividades detectadas en el reconocimiento

Bueno, aquí hay varios conceptos que necesitas entender.

Las actividades que el usuario puede realizando se representan con la clase `DetectedActivity`.

Estos objetos se caracterizan por tener un tipo y umbral de confianza.

¿Qué es tipo?

Simplemente la constante asignada para la actividad. La tabla posterior las resume:

<code>IN_VEHICLE</code>	El dispositivo va en algún vehículo
<code>ON_BICYCLE</code>	Está en una bicicleta
<code>ON_FOOT</code>	Está con un usuario que camina o corre

RUNNING	El usuario está corriendo
STILL	Sin movimiento
TILTING	Cambio brusco de inclinación del ángulo normal
UNKNOWN	No hay estado disponible
WALKING	El usuario está caminando

Limitaciones: No siempre tendrás un diagnóstico acertado. Por lo que puedes preguntar al usuario con una notificación para que confirme su actividad, antes de tomar una decisión.

Sumando, que hay dispositivos en concreto donde la API no se comporta de la mejor manera.

Pero dejando eso de lado, veamos el umbral de confianza.

¿Qué es umbral de confianza?

Entiéndelo como la seguridad de la medida. Los valores van de 0 (nada de confianza) a 100 (máxima confianza).

Nosotros solo detectaremos si el usuario está quieto, caminando o corriendo. Esto quiere decir que usaremos como filtros a: **STILL** , **WALKING** y **RUNNING** .

Manejar el Intent entrante

Ahora vamos a procesar el contenido del intent en el método **onHandleIntent()** .

Aquí está la guía para que lo hagas:

1. Obtén el resultado de reconocimiento de actividad a través de **ActivityRecognition.extractResult()** .

```
// Lo primero es obtener el resultado de reconocimiento.
```

```
ActivityRecognitionResult result = ActivityRecognitionResult.extractResult(intent);
```

2. Usa el método **getMostProbableActivity()** para obtener la actividad con más probabilidad de estar siendo realizada.

```
DetectedActivity detectedActivity = result.getMostProbableActivity();
```

3. Obtén el tipo de la activida detectada y determina si es **ON_FOOT** . Si es el caso, entonces crea un método llamado **walkingOrRunning()** .

La idea es que retorno **WALKING** o **RUNNING** , dependiendo de cuál tenga la confianza más alta.

```
...  
int type = detectedActivity.getType();
```

```
        if (DetectedActivity.ON_FOOT == type) {
            type = walkingOrRunning(result.getProbableActivities());
        }
    }

    private int walkingOrRunning(List<DetectedActivity> probableActivities) {
        int walkActivity = 0, runningActivity = 0;

        for (DetectedActivity probableActivity : probableActivities) {
            if (probableActivity.getType() == DetectedActivity.WALKING) {
                walkActivity = probableActivity.getConfidence();
            } else if (probableActivity.getType() == DetectedActivity.RUNNING) {
                runningActivity = probableActivity.getConfidence();
            }
        }

        return (walkActivity >= runningActivity) ? DetectedActivity.WALKING : DetectedActivity.RUNNING;
    }
}
```

4. Crea un nuevo **Intent** para enviar los resultados hacia **LocationActivity** .

```
Intent localIntent = new Intent(Constants.BROADCAST_ACTION);
```

La acción del intent puedes personalizarla en una clase auxiliar de constantes (**Constants**).

4. Añade el tipo al intent.

```
localIntent.putExtra(Constants.ACTIVITY_KEY, type);
```

5. Usa el **LocalBroadcastManager** para reportar el intent hacia la actividad principal.

```
LocalBroadcastManager.getInstance(this).sendBroadcast(localIntent);
```

Añadir Activity Recognition API al cliente Google

Hasta el momento nuestra app no reconoce nada.

¿Por qué?

Es necesario que iniciemos la API de reconocimiento desde algún lado, ¿no crees?

Y este lugar es **LocationActivity** .

Así que codifícalo de esta manera:

Paso #1. La API de reconocimiento de actividad requiere exactamente las mismas callbacks que la de ubicación. Debes incluirlas si tienes un proyecto que estás comenzando desde cero.

Paso #2. Crea las variables miembros para mostrar el reporte de actividad. Ya sabes que nuestro diseño tiene un **ImageView** para mostrar la actividad. Así que declaramos dicho elemento y lo obtendremos en **onCreate()** .

```
private ImageView mDetectedActivityIcon;
```

Paso #3. Añade al objeto `GoogleApiClient` la api de reconocimiento de actividad con la constante `ActivityRecognition.API`.

```
private synchronized void buildGoogleApiClient() {
    mGoogleApiClient = new GoogleApiClient.Builder(this)
        .addConnectionCallbacks(this)
        .addOnConnectionFailedListener(this)
        .addApi(LocationServices.API)
        .addApi(ActivityRecognition.API)
        .enableAutoManage(this, this)
        .build();
}
```

Crear Broadcast Receiver para obtener actividades detectadas

Puedes recibir los intents que vienen desde `DetectedActivitiesIntentServices` agregando un componente `BroadcastReceiver`.

Pasemos al código para llevarlo a cabo:

Paso #1. Crea una clase anidada llamada `ActivityDetectionBroadcastReceiver` que extienda de `BroadcastReceiver`.

```
public class ActivityDetectionBroadcastReceiver extends BroadcastReceiver{
```

Paso #2. Sobrescribe su método `onReceive()`.

```
public class ActivityDetectionBroadcastReceiver extends BroadcastReceiver{

    @Override
    public void onReceive(Context context, Intent intent) {

    }
}
```

Paso #3. Implementa `onReceive()`.

Lo primero es obtener el tipo de actividad.

En otras palabras, usa el método `getIntExtra()` del intent que entra como parámetro.

```
@Override
public void onReceive(Context context, Intent intent) {
    int type = intent.getIntExtra(Constants.ACTIVITY_KEY, -1);

    mImageResource = getActivityIcon(type);
    updateRecognitionUI();
}
```

Paso #4. Poner icono de actividad.

Dependiendo del tipo de actividad detectada, así mismo usaremos un ID drawable para el image view.

Creemos un método para ello.



Llámalo `getActivityIcon()` y en su interior usa un switch con los tipos y los IDs.

```
private int getActivityIcon(int type) {
    switch (type) {
        case DetectedActivity.STILL:
            return R.drawable.ic_still;
        case DetectedActivity.WALKING:
            return R.drawable.ic_walk;
        case DetectedActivity.RUNNING:
            return R.drawable.ic_run;
        default:
            return R.drawable.ic_question;
    }
}
```

Paso #5. Crea una nueva instancia del broadcast receiver en `onCreate()` .

```
mBroadcastReceiver = new ActivityDetectionBroadcastReceiver();
```

Administrar el cliente

Ya hemos visto que la API de reconocimiento de actividad requiere pasos similares a la de ubicación.

Por esta razón... el proceso es similar:

Paso #1. Si no usas `enableAutoManage()` en el cliente, entonces conecta y desconecta en `onStart()` y `onStop()` .

Paso #2. Registra el broadcast receiver en `onResume()` con un `IntentFilter` , cuya acción sea la misma que usaste en `DetectedActivitiesIntentService` .

Recuerda usar el método `LocalBroadcastManager.registerReceiver()` .

```
IntentFilter intentFilter = new IntentFilter(Constants.BROADCAST_ACTION);
LocalBroadcastManager.getInstance(this)
    .registerReceiver(mBroadcastReceiver, intentFilter);
```

Paso #3. Completa el ciclo registro/desregistro en `onPause()` . De forma similar, usa `LocalBroadcastManager.unregisterReceiver()` .

```
LocalBroadcastManager.getInstance(this)
    .unregisterReceiver(mBroadcastReceiver);
```

Iniciar reconocimiento de actividades

¡Por fin!

Luego de todo ese código, vamos a ver como iniciar el monitoreo.

Mantente atento:

Paso #1. Ve al método `onConnect()` y llama a `ActivityRecognition.ActivityRecognitionApi.requestActivityUpdates()` .



¿Ves que el formato es muy similar al de las actualizaciones de ubicación?

Para que funcione, pásale el cliente, un intervalo de actualización y una referencia futura del servicio que creaste.

```
ActivityRecognition.ActivityRecognitionApi.requestActivityUpdates(  
    mGoogleApiClient,  
    Constants.ACTIVITY_RECOGNITION_INTERVAL,  
    getActivityDetectionPendingIntent()  
).setResultCallback(this);
```

Si deseas aumentar la reusabilidad de esta tarea, entonces sepárala en un nuevo método. Igualmente puedes hacerlo con el PendingIntent.

```
private void stopActivityUpdates() {  
    ActivityRecognition.ActivityRecognitionApi.removeActivityUpdates(  
        mGoogleApiClient,  
        getActivityDetectionPendingIntent()  
    ).setResultCallback(this);  
}  
  
private PendingIntent getActivityDetectionPendingIntent() {  
    Intent intent = new Intent(this, DetectedActivitiesIntentService.class);  
    return PendingIntent.getService(this, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT);  
}
```

Paso #2. Programa el ciclo pausa/reanudación para las actualizaciones de actividades.

Inicia el reconocimiento en **onResume()** si el cliente está conectado.

Detén el reconocimiento en **onPause()** con el método **ActivityRecognition.ActivityRecognitionApi.removeActivityUpdates()** .

```
@Override  
protected void onResume() {  
    super.onResume();  
    if (mGoogleApiClient.isConnected()) {  
        startLocationUpdates();  
        startActivityUpdates();  
    }  
  
    IntentFilter intentFilter = new IntentFilter(Constants.BROADCAST_ACTION);  
    LocalBroadcastManager.getInstance(this)  
        .registerReceiver(mBroadcastReceiver, intentFilter);  
}
```

```
@Override  
protected void onPause() {  
    super.onPause();  
    if (mGoogleApiClient.isConnected()) {  
        stopLocationUpdates();  
        stopActivityUpdates();  
    }  
  
    LocalBroadcastManager.getInstance(this)  
        .unregisterReceiver(mBroadcastReceiver);  
}
```

Volvemos a lo mismo. Crea un método para representar como unidad conceptual la detención de actualizaciones.

```
private void stopActivityUpdates() {  
    ActivityRecognition.ActivityRecognitionApi.removeActivityUpdates(  
        mGoogleApiClient,  
        getActivityDetectionPendingIntent()  
    ).setResultCallback(this);  
}
```


Procesar resultados con ResultCallback

Resulta que cuando usas request/remove para el reconocimiento de actividades, requieres procesar el desenlace de sus llamadas.

He aquí en pocas palabras el procedimiento...

Paso #1. Implementa en la actividad la interfaz `ResultCallback<Status>` .

Paso #2. Sobrescribe el método `onResult()` .

```
@Override
public void onResult(@NonNull Status status) {

}
```

Paso #3. Decide qué acciones tendrá. En mi caso solo loguearé tanto el resultado positivo como el negativo.

```
@Override
public void onResult(@NonNull Status status) {
    if (status.isSuccess()) {
        Log.d(TAG, "Detección de actividad iniciada");
    } else {
        Log.e(TAG, "Error al iniciar/remover la detección de actividad: "
            + status.getStatusMessage());
    }
}
```

Corre la aplicación final

Finalmente.

Corre toda la app que llevamos hasta el momento y verifica los comportamientos.

Te recomiendo que uses tu dispositivo real para testear el reconocimiento de actividad

Crea Tu Propia App Con Localización

Ha llegado el momento de que crees tu propia app con ubicación y reconocimiento.

Este artículo es solo el comienzo para manejar estas APIs.

Por el momento ya hemos visto varios tutoriales avanzados que puedes complementar como [Google Maps](#), [Firebase Cloud Messaging](#), [AdMob](#), etc.

Incluso si quieres investigar más de integrar varias APIs de Google relacionadas al contexto del usuario (entre ellas las dos que vimos aquí), lee sobre [Google Awareness API](#). Mira el link por si te interesa.



Find Cheapest Money Transfer

Anuncio Monito

Tutorial De Bases De Datos SQLite...

hermosaprogramacion.com

Invierte en personas

Anuncio Afluenta

Crear Logi Android Co

hermosaprogram

Sledovanie vozidiel

Anuncio sledovanie-vozidiel.sk

App Productos 3: Detalle De...

hermosaprogramacion.com

Libreria Retrofit En Android Parte 3:...

hermosaprogramacion.com

Retrofit Er Parte 4: Cr

hermosaprogram

45 Comentarios

Hermosa Programación

1 Acceder

Recomendar 8

Tweet

Compartir

Ordenar por los mejores



Únete a la conversación...

INICIAR SESIÓN CON

O REGISTRARSE CON DISQUS

Nombre



José • hace 3 años

Me imagino todo el tiempo y la dedicación que te lleva hacer todo esto. No dudo que tus tutoriales sean de lo mejor que se encuentra en español. Un saludo.

5 | • Responder • Compartir



James Moderador José • hace 3 años

Me alegra leer tu comentario José, de verdad me esfuerzo para mis lectores entiendan. Saludos!

2 | • Responder • Compartir



neo • hace 3 años

genial james!!!! eres el mejor!!!! se nota el detalle y el tiempo que dedicas a tus trabajos.

2 | • Responder • Compartir



James Moderador neo • hace 3 años

Gracias Neo, ojala te sirva!

| • Responder • Compartir



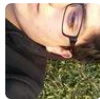
Juan Ignacio Alvarez • hace 2 años

Hola James. Una duda:

"A continuación, ubícate en onCreate() y crea la instancia de GoogleApiClient." tu pones eso en el tuto pero en el código que bajé no sale esa declaración en onCreate... por qué es eso??? y otra duda:
mLastLocation =
LocationServices.FusedLocationApi.getLastLocation(mGoogleApiClient); me tira un error de "code should explicitly check if permission is available"

Está muy bien el tutorial, muchas gracias de ante mano

1 | • Responder • Compartir



Steve • hace 3 años

Eres un capo James, como comentan abajo tus artículos son de lo mejor que se encuentra en español.

1 | • Responder • Compartir




James Moderador Steve • hace 3 años

Gracias mi hermano!



| • Responder • Compartir




ty • hace 3 años





excelente tuto


1  |  • Responder • Compartir ›



James Moderador ➔ ty • hace 3 años

Gracias compañero!

 |  • Responder • Compartir ›





Marco Antonio • hace 3 años

Genial!!!

una consulta como se podria mostrar las mismas ubicaciones en Google Maps...desde la aplicacion..



Saludos, y gracias por el tutorial.


1  |  • Responder • Compartir ›



James Moderador ➔ Marco Antonio • hace 3 años

Hola marco, ese es un tuto para el futuro. Saludos!



 |  • Responder • Compartir ›




webserveis • hace 3 años • edited

Vaya currada de tutorial, se agradece un motón el tiempo que te tomas en realizas esos pedazos de tutoriales.



Me viene de fabula ya que estoy aprendiendo ese aspecto realizando un app estilo "find my car"


1  |  • Responder • Compartir ›



James Moderador ➔ webserveis • hace 3 años



Gracias compañero webserveis. Si claro, podría serte de utilidad. Saludos!


 |  • Responder • Compartir ›



Offer J. Mosquera C. • hace 3 años



Buenas noche, James excelente material . . . me preguntaba si puedes compartir el código fuente de la app. Mil gracias.


1  |  • Responder • Compartir ›



James Moderador ➔ Offer J. Mosquera C. • hace 3 años



Gracias Offer. Si claro, siempre los comparto. Está en la parte inicial en la caja amarilla, cerca al video


 |  • Responder • Compartir ›



Offer J. Mosquera C. ➔ James • hace 3 años



James, Muchas gracias . . . excelente material.


1  |  • Responder • Compartir ›



Leo Zeron • hace 3 años



Excelente


1  |  • Responder • Compartir ›



James Moderador ➔ Leo Zeron • hace 3 años



Gracias Leo!


 |  • Responder • Compartir ›



Jesús Flor Farias • hace 3 años



Hace poco que comienzo a seguir tu pagina, y déjame decirte que son unos artículos estoy aprendiendo bastante. Gracias James.


1  |  • Responder • Compartir ›



James Moderador ➔ Jesús Flor Farias • hace 3 años



Dale Jesús, muchas gracias. Sigue atento, vendrán más.


 |  • Responder • Compartir ›



Julio Vergara • hace 3 años



Excelente James.....sólo falta uno de geofences y puntos de interés... :)

1  |  • Responder • Compartir ›



James Moderador ➔ Julio Vergara • hace 3 años

Gracias Julio, claro, esos temas son vitales también.

 |  • Responder • Compartir ›



Jhon Freddy Gómez Franco • hace un año





Hola,

Quisiera saber si desde un dispositivo aparte gps le envío datos a mi aplicación y este los muestre en mi mapa creado en la aplicación, com haría para ingresar esos datos??
Puedes guiarme?? gracias

^ | v • Responder • Compartir ›



Alejandro Ernesto Lora Carrill • hace 2 años

muy bueno el totu, alguien me podría pasar el proyecto ya que no puedo descargarlo...gracias

^ | v • Responder • Compartir ›



Jesus Villa Sanchez • hace 2 años

James una consulta...y si el desactiva el GPS y ya lo habia prendido...como hace para que siempre lo obligue a tener prendido el GPS con el callback que te muestra ese dialogo de gps...

^ | v • Responder • Compartir ›



Vale Mariel Sanchez Torrico • hace 2 años

James me encanto el tutorial muy bien explicado 100 funcional..
Una consulta como puedo mandar estos datos para que se muestren en un mapa?

^ | v • Responder • Compartir ›



Julio Vergara • hace 2 años

Hola James....sobre este tema...harás más? de GeoFences, puntos de interés etc? me apasiona la geolocalización y los mapas.....Gracias y sigue así...el mejor sitio de tutos de Android por mucho...saludos desde mi querida y maltratada Venezuela...!!!

^ | v • Responder • Compartir ›



Christian Sari • hace 3 años

Gracias James, buen tutorial, estaba tratando de implementarlo pero no puedo descargar el codigo para revisar errores que tengo, al ingresar mi correo en la casa me sale un mensaje "Error check".

Espero puedas solucionarlo.

Saludos y éxitos.

^ | v • Responder • Compartir ›



Victor Alvarado • hace 3 años

Hola James! Excelente tutorial, gracias por compartir tus conocimientos. Te quería pedir un consejo: estoy creando un App para monitorear la ubicación pero no he conseguido obtener el Location si el GPS está desactivado. O sea necesito saber la ubicación independientemente si el usuario enciende o apaga su gps. Hay forma de hacerlo , algún consejo?

^ | v • Responder • Compartir ›



Aldo Rodrigo Sánchez González • hace 3 años

OnRequestPermissionsResult(...) es invocada cuando se llama
ActivityCompat.requestPermissions pero solo funciona si el primer método funcionado está dentro de una actividad como lo expones en este ejemplo. ¿Qué ocurre para los fragmentos (v4.app.Fragment)? Estoy leyendo que se usa solamente requestPermissions sin ActivityCompat pero no he tenido éxito. Saludos

^ | v • Responder • Compartir ›



mwtronic • hace 3 años • edited

Saludos. gracias por la información... esta bastante interesante.
Tengo una duda.... Un caso de que yo necesite consultar las coordenadas GPS desde dos o mas "Activity", que me recomendaría para no repetir el mismo código en cada actividad? Gracias.

^ | v • Responder • Compartir ›



Victor Alvarado ➔ mwtronic • hace 3 años

Yo necesito hacer lo mismo, acceder a la ubicación desde distintos fragments.
Cómo se puede hacer en ese caso. De antemano, muchas gracias.

^ | v • Responder • Compartir ›




jorgecoronado • hace 3 años

el link de descarga no funciona

^ | v • Responder • Compartir ›




 **jorgecoronado** ➔ jorgecoronado • hace 3 años

[fancy_box id=5 linked_cu=5940]Haz click para descargar el código completo.


[/fancy_box]

⤴ | ⤵ • Responder • Compartir ›

 **James** Moderador ➔ jorgecoronado • hace 3 años


Desactivé el plugin de descargas,estoy en ello para que los links funcionen de nuevo. Dame algo de tiempo

⤴ | ⤵ • Responder • Compartir ›

 **Itzli Molina** • hace 3 años


Saludos y gracias por este excelente curso. Una duda, no esta activo el link para descargar el codigo, tengo unas dudas y creo que el codigo me seria de mucha ayuda. Gracias de antemano

⤴ | ⤵ • Responder • Compartir ›

 **José** • hace 3 años


Hola, qué excelente tutorial! Tengo una pregunta, para recuperar la latitud y longitud es necesario siempre tener activa la localización en android?

⤴ | ⤵ • Responder • Compartir ›

 **Juan Carlos Jirón** • hace 3 años


James, la verdad no tengo palabras para describir tu gran trabajo, es impecable, y no dudo en decirlo, este blog es el mejor que hay en internet para aprender a desarrollar android en México, muchas gracias

⤴ | ⤵ • Responder • Compartir ›

 **Daniel Alejandro Garcia** • hace 3 años


Una duda, el api te brinda alguna utilidad para detectar la proximidad con algún punto en el mapa por ejemplo que me digo que estoy a x metros de mi restaurante favorito.

⤴ | ⤵ • Responder • Compartir ›

 **Jorge Iván Martínez Puello** • hace 3 años

Interesante, muy bueno, tengo una duda, es una sola apalicación para el cliente como para el monitor?


⤴ | ⤵ • Responder • Compartir ›

 **webserveis** • hace 3 años

Una pregunta sobre los IntentSercices, veo que cuando se les asigna al manifest, solo instalar la app, se ejecuta, sin necesario de abrirla, digamos consumiendo recursos para nada.


Se puede trasladar hacia un services, que se puede incializar y parar?

⤴ | ⤵ • Responder • Compartir ›

 **Kaploc** • hace 3 años


Hola que tal?? de ante mano gracias por los tutoriales me han servido, yo tengo una pregunta, estoy haciendo una app parecida cliente/servidor lo que quiero hacer es que los clientes puedan observar a la ubicacion de una persona y que le permita llamar, tipo whatsapp, mi pregunta es como se hacen las llamadas mediante internet???

⤴ | ⤵ • Responder • Compartir ›

 **Marco Antonio** • hace 3 años


Hola, excelente tutorial, tengo una duda como se podría mostrar esas ubicaciones en un Mapa (Google Maps)

⤴ | ⤵ • Responder • Compartir ›

 **Angel Uc** • hace 3 años

cual es la diferencia entre usar el location de android y el location de google?

⤴ | ⤵ • Responder • Compartir ›

 **James** Moderador ➔ Angel Uc • hace 3 años

Hola Angel, en el artículo "Location Strategies" (<https://developer.android.c...> del framework, Google nos da la explicación: (ver imagen)

En resumen, obtener ubicaciones con Play Services nos da más nivel de automatización, precisión y manejo personalizado del consumo de energía.

Saludos amigo!





^ | v • Responder • Compartir ›

TAMBIÉN EN HERMOSA PROGRAMACIÓN


App Productos 4: Creación De Facturas

35 comentarios • hace 2 años

 **Jerson Granados** — Hola amigo James, si [Avatar](#) compré la parte 3 tengo algun descuento?

CheckBox: Controles En Android

11 comentarios • hace 4 años

 **Gabriel B.** — Hola James, gracias por [Avatar](#) responder, ya resolví el problema, lastimosamente tuve que hacerlo con

Premio Top 80 Mobile App Blogs de Feedspot



Hermosa Programación: [+50 Tutoriales Desarrollo Android](#) Copyright © 2019.

